Utiliser un formulaire

L'un des points forts de PHP est sa capacité à gérer les formulaires. Le concept de base qui est important à comprendre est que tous les champs d'un formulaire seront automatiquement disponibles dans le script PHP d'action. Lisez le chapitre du manuel concernant les variables depuis des sources externes à PHP pour plus d'informations et d'exemples sur la façon d'utiliser les formulaires. Voici un exemple de formulaire HTML :

Exemple #1 Un simple formulaire HTML

```
<form action="action.php" method="post">
 <p>Votre nom : <input type="text" name="nom" /></p>
 <p>Votre âge : <input type="text" name="age" /></p>
 <p><input type="submit" value="OK"></p>
</form>
```

Il n'y rien de particulier dans ce formulaire. Il est en HTML pur, sans aucune configuration particulière. Lorsque le visiteur remplit le formulaire, et clique sur le bouton OK, le fichier `action.php` est appelé. Dans ce fichier, vous pouvez écrire le script suivant :

Exemple #2 Afficher des données issues d'un formulaire

```
Bonjour, <?php echo htmlspecialchars($_POST['nom']); ?>.
Tu as <?php echo (int)$_POST['age']; ?> ans.
```

Voici le résultat que vous pourriez obtenir, selon les valeurs que vous avez saisies :

```
Bonjour Jean.
Tu as 29 ans.
```

Mise à part les parties htmlspecialchars() et *(int)*, ce script ne fait que des choses évidentes. htmlspecialchars() s'assure que tous les caractères spéciaux HTML sont proprement encodés afin d'éviter des injections de balises HTML et de Javascript dans vos pages. Pour l'âge, vu que nous savons que c'est un entier, vous pouvez le convertir en un entier. Vous pouvez également demander à PHP de le faire automatiquement à votre place en utilisant l'extension filter. Les variables `$_POST['nom']` et `$_POST['age']` sont automatiquement créées par PHP. Un peu plus tôt dans ce tutoriel, nous avons utilisé la variable `$_SERVER`, une superglobale. Maintenant, nous avons introduit une autre superglobale `$_POST` qui contient toutes les données envoyées par la méthode POST. Notez que dans notre formulaire, nous avons choisi la *méthode* POST. Si vous avions utilisé la méthode *GET* alors notre formulaire aurait placé ces informations dans la variable `$_GET`, une autre superglobale. Vous pouvez aussi utiliser la variable `$_REQUEST`, si vous ne souhaitez pas vous embarrasser de la méthode utilisée. Elle contient un mélange des données de GET, POST, COOKIE et FILE.

Vous pouvez également utiliser des champs XForms dans PHP, même si vous vous sentez bien avec l'utilisation des formulaires HTML. Bien que le travail avec XForms ne soit pas fait pour les débutants, vous pourriez être intéressé par cette technologie. Nous avons également une courte introduction sur le traitement des données reçues par XForms dans notre section sur les fonctionnalités.

User Contributed Notes 4 notes

add a note

104 sethg at ropine dot com 10 years ago

According to the HTTP specification, you should use the POST method when you're using the form to change the state of something on the server end. For example, if a page has a form to allow users to add their own comments, like this page here, the form should use POST. If you click "Reload" or "Refresh" on a page that you reached through a POST, it's almost always an error -- you shouldn't be posting the same comment twice -- which is why these pages aren't bookmarked or cached.

You should use the GET method when your form is, well, getting something off the server and not actually changing anything. For example, the form for a search engine should use GET, since searching a Web site should not be changing anything that the client might care about, and bookmarking or caching the results of a search-engine query is just as useful as bookmarking or caching a static HTML page.

18 Johann Gomes (johanngomes at gmail dot com) 3 years ago

Also, don't ever use GET method in a form that capture passwords and other things that are meant to be hidden.

11 Andy 2 years ago

Just a reminder about security: the chosen HTTP request verb (e.g. POST, GET) does not necessarily make the request "secure". Any information that is not transmitted over an encrypted channel (using SSL, i.e. HTTPS) is transmitted in plan text.

For secure transport of sensitive/private information over HTTP consider using SSL as this prevents eve's dropping of the information transmitted over HTTP.

[Edited by googleguy@php.net for clarity]

-55 behnam jaza faza 10 months ago

The note above says:

"Also see the import_request_variables() function. "

But dont:

This function has been DEPRECATED as of PHP 5.3.0 and REMOVED as of PHP 5.4.0.

add a note

Une introduction à PHP

Le nécessaire
Votre première page PHP
Trucs pratiques
» **Utiliser un formulaire**
Utiliser des codes anciens avec les nouvelles versions de PHP
Et après ?

Copyright © 2001-2014 The PHP Group My PHP.net Contact Other PHP.net sites Mirror sites

Privacy policy